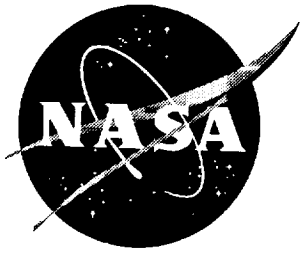
NASA/CR-1998-206907



The LVLASO I/O Concentrator Software Description, Version 3.5

Christopher J. Slominski, Valerie E. Plyler, and David A. Wolverton
Computer Sciences Corporation, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-20431

January 1998

Available from the following:

NASA Center for AeroSpace Information (CASI)
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

Section 1.0 System Description

Overview

The software described in this document resides in the I/O concentrator unit of the Low Visibility Landing and Surface Operations (LVLASO) experimental configuration used on-board NASA's Boeing 757-200 aircraft. The I/O concentrator has several functions. It serves as a central point for on-board data buses, provides a layer of protocol for the LVLASO radio datalinks, computes Global Positioning Satellite (GPS) navigation parameters, and interfaces to on-board data recording. The I/O concentrator is a VME chassis with a Motorola PowerPC MVME-1604 processor board in slot #0, serving as the bus master. The remaining boards populating the chassis include a Systran SCRAMNet+, Condor Engineering ARINC-429, VMIVME-2128 discrete output, VMIVME-1129 discrete input, Datum bc635VME time processor, and a VMIVME-3122 Analog-to-Digital converter. The CPU on the processor board is a 133 MHz MPC604 PowerPC having 32 Mbytes of memory. The real-time operating system is VxWorks 5.3. The operating system and applications code are bundled together and stored in *Flash RAM*. The bundled code is loaded into PowerPC RAM at boot up.

The I/O concentrator receives inputs from Boeing 757 data buses, a Collins GPS receiver, Collins radio datalinks, and the pilot input device (PID). It continually updates parameters on the SCRAMNet+ ring for use by the Silicon Graphic Inc. (SGI) workstations and the Data Acquisition System (DAS). It also can receive requests for downlink acknowledgments and cockpit lighting commands from the workstations on the ring (figure 1.0). The ship's systems inputs consist of 9 asynchronous ARINC-429 buses and three analog sources for body mounted accelerations (longitudinal, lateral, and normal). A variety of ship's data are brought in on the ARINC-429 buses, at rates ranging from 1 Hz to 50 Hz (Appendix A). These parameters are decoded for use in internal navigation calculations and for output to the other nodes on the SCRAMNet+ ring. The GPS receiver outputs its data also on an ARINC-429 bus. The GPS data is blended with the aircraft's Inertial Reference Unit's (IRU) data to obtain a continuous real-time navigation solution. Both types of datalink inputs are provided to the I/O concentrator as ARINC-429 data buses. Sets of ARINC-429 words are combined to form datalink packets, which are decoded and passed to the SGI nodes as messages from ground controllers. Similarly, datalink transmissions from the I/O concentrator are sent via ARINC-429. The PID is a simple device which has dials and switches used by the pilot to control the LVLASO displays and to acknowledge Air Traffic Control (ATC) instructions. It has two lights which are driven by the SGIs. The I/O concentrator interfaces to the PID hardware via discrete lines.

Operator Control

The operation of the software depends on which type of Flash-RAM configuration is installed on the MVME-1604 processor board. During development, the Flash-RAM contains only a small subset of VxWorks and no applications software. When the processor board is reset, the "mini-kernel" is loaded into processor memory and executed. The "mini-kernel" goes to a predetermined location on a host computer and downloads the full operating system via Ethernet. The developer may then download versions of applications software using the Wind Rivers tool called the WindShell. Alternately, the applications software may be bundled with the full operating system and loaded on Flash-RAM. In this stand-alone configuration no Ethernet is used. All software is obtained from Flash-RAM at boot time. This scenario is used when the applications software is complete and is being used in its intended environment. Refer to Wind Rivers Corp. documentation for information on making the various configurations of the VxWorks operating system.

To use the I/O Concentrator software in the development configuration, the operator must initiate the software by entering "*sp run*" to the WindShell prompt. The *monitor* program is also started at the WindShell prompt by entering "*monitor*". In the stand-alone configuration the I/O concentrator tasks, except the *monitor*, are started automatically after the operating system boots. The operating system's source code was modified to include a spawn of the *run* task after it completes its initialization. Once again, the *monitor* is started by entering "*monitor*" to the target shell prompt appearing on the PowerPC console.

There are a few global parameters which may be modified by the operator to select optional I/O concentrator configurations. These selections may be on either the development or stand-alone

configuration. The shell (WindShell or target shell) allows the assignment of integer values to global parameters as follows; **-> variable_name = new_value**. The current setting of a parameter may be checked by typing the name of the parameter at the shell prompt. The following table contains the software configuration parameters and the various settings.

Parameter	Default	Description
LabFlg	0	enables/disables lab reset of the navigation position; 0-aircraft 1-lab
BmaFlg	0	Accelerometer selection; 0-IRU accelerometers 1-Body mounted accelerometers
TestNum	0	SCRAMNet test selection; 0-incrementing pattern 1-rotating pattern 2-static engineering values
ValidCnt	750	controls the length of the DGPS coasting period (1 unit = 1/25 seconds)

Figure 1. Configuration variables.

Software Configuration

The applications software running on the PowerPC is written in the 'C' programming language. It is partitioned into a number of VxWorks tasks which are independent threads of execution. The following table lists the tasks involved with the LVLASO applications.

Task name	*Type*	*Priority*
Run	system startup	
t50hz	cyclical-50Hz	20
t25hz	cyclical-25Hz	50
AtcRecv	event driven	100
AtcXmit	event driven	100
AmassRecv	event driven	100
Background	cyclical-background	150
Monitor	interactive utility	200

Figure 2. I/O Concentrator tasks.

In addition to the applications tasks, there are two pieces of software which do not run in a task context, but are activated at the interrupt level. One is attached to the VxWorks clock interrupt handler for applications scheduling, and the other is the VME interrupt handler used for asynchronous event management on the SCRAMNet+ network ring.

Task: *run*

This task is spawned at system start up, by either the VxWorks operating system in a stand-alone configuration or manually by the I/O concentrator operator in a development configuration. It is responsible for setting up the software environment and spawning the other (non-utility) tasks. When *run* has completed setting up, it exits and does not become active during the normal operation of the I/O concentrator. The setup operations include:

- Set target shell priority (stand-alone configuration only).
- Set system clock interrupt rate (50 Hz).
- Map VME I/O boards to local processor address space.
- Configure SCRAMNet interrupt handling.
- Configure ARINC-429 receiver and transmitter channels.
- Set up operating system data structures used in inter-task communications.
- Spawn applications tasks.
- Attach clock interrupt handler.
- Arm hardware reboot watchdog.

Task: *t50hz*

This task is the main real-time applications task. It runs at the highest priority of all applications and is allowed to repeat its real-time cycle once every 20 milliseconds by the clock interrupt handler (*tick_mgr*). *t50hz* performs two operations in its cycle. First it tests each ARINC-429 receiver queue. If a queue has received ARINC words they are removed one at a time (up to a maximum of 25) and passed to the decoding functions. The second operation performed is the sampling of the analog input channels, which contain the outputs from the body mounted accelerometers. The input accelerations are filtered to remove noise.

Task: *t25hz*

This task is a real-time cyclical task, repeating its operations once every 40 milliseconds. It operates at a higher priority than the other applications tasks, except for *t50hz*. The operations performed within each real-time cycle inlcude:

- Sample the UTC clock from the Datum time processor board.
- Perform GPS navigation calculations.
- Store parameters in data recording buffers.
- Perform discrete input/outputs for Pilot Interface Device (PID).
- Perform miscellaneous SCRAMNet output calculations.

Task: *atcRecv*

This is an event driven task operating at a priority below the real-time tasks, and above background and utility tasks. It responds to ATC uplink messages received by the I/O concentrator. It parses the uplink message and translates the information into ASCII text controller instructions for the flight crew. The translated message is stored in SCRAMNet memory for access by the LVLASO SGI workstations. When it finishes with a message, it suspends itself to await another ATC uplink. Raw uplinks are first processed by *t50hz* to form a complete uplink packet, removing all of the radio control data from the packets. The resulting binary data stream is what is queued to *atcRecv*.

Task: *atcXmit*

This is an event driven task operating at a priority below the real-time tasks, and above background and utility tasks. It responds to downlink messages becoming available from a LVLASO workstation. The workstation interrupts the I/O concentrator when it has stored downlink parameters in shared SCRAMNet memory. The I/O concentrator interrupt handler fetches the parameters and queues them to *atcXmit*, which encodes the downlink into binary data fields. *atcXmit* then formats the data fields into the ARINC-429 transmission packet format and queues the resulting words into an ARINC-429 transmitter queue.

Task: *amassRecv*

This is an event driven task operating at a priority below the real-time tasks, and above background and utility tasks. It responds to AMASS/Traffic uplink messages received by the I/O concentrator. It

parses the uplink message and stores the data fields in SCRAMNet memory for access by the LVLASO SGI workstations. When it finishes with a message, it suspends itself to await another uplink. Raw uplinks are first processed by *t50hz* to form a complete airport scan, removing all of the radio control data from the packets. The resulting binary data stream is what is queued to *amassRecv*.

Task: *background*

This task runs continually, independent of clock interrupts and operating system events. It is lower priority than all tasks except the *monitor* utility. It performs its operations in a continuously repeated cycle, pausing a fixed delay at the end of each cycle to let the *monitor* have some execution time. The operations performed each cycle include:

- Invalidate ARINC-429 parameters which have not been received within their required update frequency.
- Extract current number of SCRAMNet nodes and inhibit SCRAMNet error interrupts from occurring more often than once each second.
- Check status of the two real-time tasks. If both are good, reset hardware reboot watchdog to avoid an automatic reboot.
- Delay 5 clock ticks (0.1 seconds) for the *monitor* utility

Task: *monitor*

This task is the sole I/O concentrator utility program. It operates at a priority lower than all the applications tasks and higher than the target command shell (stand-alone configuration only). It utilizes the I/O concentrator screen to display specifically selected global variables. See "Appendix C Using the Monitor Program" for information on usage of the *monitor* utility.

Interrupt Handler: *tick_mgr*

This module is attached to the VxWorks clock interrupt handler. It is called by the clock interrupt handler every clock tick, with 50 clock ticks occurring each second. Each time it is called it places the real-time task *t50hz* in the "ready to run" queue. The real-time task *t25hz* is queued every other clock tick. On each activation of the handler where *t25hz* is not called, *tick_mgr* increments the SCRAMNet system synchronization word. The system synchronization word is configured to cause SCRAMNet memory interrupts on other nodes attached to the ring.

Interrupt Handler: *memAction*

This module is attached to the VME interrupt vector 254 at interrupt priority level 4. It is called whenever a SCRAMNet memory interrupt is received, and is passed the memory offset of the SCRAMNet location causing the interrupt. There are two SCRAMNet memory interrupts expected. Any other offset will be counted as an error. One interrupt occurs when a node on the ring signals for a downlink packet transmission. In this situation the downlink data is stored in a data recording holding area and also placed in an operating system message queue to activate the *atcXmit* task. The other type of interrupt occurs when a node on the ring signals that all of the parameters in the ROTO data area have been computed and are ready to be sampled as a complete set. The ROTO data is stored in a recording holding area until the data recording software in *t25hz* is ready to use it.

Section 2.0 VME I/O

The heart of the I/O concentrator is the VME bus. The I/O concentrator chassis has seven cards attached to the VME bus. The following sections describe the various cards, and how they are utilized by the I/O concentrator software running on the MVME-1604 processor card.

Space	Size	Local Addresses	Bus Addresses
A16	64K	EFFF0000 - EFFFFFFF	0000-FFFF
A24	16M	E0000000 - E0FFFFFF	000000 - FFFFFF
A32	16M	D8000000 - D8FFFFFF	08000000 - 08FFFFFF

Figure 3. VME address space.

Device	Addresses	Size
SCRAMNet	E000 - E03F	64 Bytes
VMIVME - 2128	E040 - E05F	32 Bytes
VMIVME - 1129	E060 - E07F	32 Bytes
bc635VME	E080 - E0BF	64 Bytes
VMIVME - 3122	F000 - FFFF	4 Kbytes

Figure 4. A16 space usage.

Device	Addresses	Size
VME-429	08000000 - 083FFFFF	4 Mbytes
SCRAMNet	08400000 - 085FFFFF	2 Mbytes

Figure 5 - A32 space usage.

Motorola MVME-1604 Processor

The first slot of the VME chassis contains the processor card, which also serves at the bus master. This card has a PowerPC 604 Central Processing Unit (CPU) running at 133 Mhz. There are 32 Megabytes of on-board memory. The processor runs the real-time operating system VxWorks (version 5.3). It also runs all the applications software described in this document. The operating system manages the VME bus by setting up mapping registers and handling bus interrupts. The applications software uses the operating system mapping to access the VME space via address pointers. The range of VME addresses mapped into the processor is shown in "Figure 3. VME address space". Applications interrupt service routines are actually just extensions to the operating system interrupt service routines. The applications software attaches its functions to the operating system's service routines, keeping the low level processor specific interrupt handling within the operating system.

Condor ARINC-429

The I/O Concentrator performs all ARINC-429 serial I/O through the Condor ARINC-429 board. The ARINC-429 board is an intelligent I/O device with its own on-board processor. The applications software running on the MVME-1604 communicates with the ARINC-429 processor by addressing memory mapped data structures associated with the board. The data structures are mapped into VME A32 space using the address range 08000000-083FFFFF. One of the data structures is the processor's System Command Queue (SCQ) which is used to request specific actions from the on-board ARINC-429 processor. The board features 16 receiver channels and 8 transmitter channels. See "Appendix A

ARINC-429 Receiver Channels" for a list of receiver assignments. For detailed information on the Condor ARINC-429 board, refer to the following.

VME-429
User's Manual (Rev 1.15)
Condor Engineering Inc., (805) 965-8000
support@condoreng.com

The applications software task *run* configures the Condor ARINC-429 board by executing the function *init_429*. The single transmitter channel used (channel #0) is configured for Air Traffic Control downlinks. Details of the datalink operations are described in "Section 3.0 Datalink Processing". Receiver channels not used in datalinks are all configured by calls to the function *arm_recv*. A call to this function is made for each ARINC-429 parameter required to be received. The function manipulates the Condor on-board data structures to allow a particular channel to accept input of a given ARINC-429 label with a specific SDI value. In essence this creates a 10 bit label for unique identification of ARINC-429 data parameters. Data structures in the applications software are also generated by calls to *arm_recv*. A linked list is created for each receiver channel, containing information about each input parameter which is armed on that channel. The information includes the label, Source Destination Index (SDI) value, decoding information, validity flag, and a time-out value. The decoding information can be simply the address of a special function that is called to process the incoming parameter, or the information needed to convert the input to an engineering unit value which will be stored as a program variable. Once the linked lists for the channels are finished, setup of the ARINC-429 board is complete and the Condor ARINC-429 on-board processor is commanded to go to its active state.

The Condor ARINC-429 on-board processor adds 32-bit ARINC-429 words to its circular queues, which are maintained for each channel, as they are received. Each circular queue can hold a maximum of 10,000 32-bit ARINC-429 words. The Inputting of the ARINC-429 parameters to the applications software on the PowerPC is performed by the real-time task *t50hz*. This task checks the on-board data structures of the Condor card every 20 milliseconds and extracts 32-bit words received on all of the queues since the last cycle. Each parameter extracted is passed to the function *decode_429* which performs a binary search on the linked list data structure belonging to the channel the parameter was received on. The decoding information stored in the list for the 10-bit label/SDI of the input parameter is used to determine the action required to decode the 32-bit ARINC-429 word. If a special function was designated, it is called immediately to perform custom decode operations. Otherwise the status matrix bits are tested to determine validity of the parameter. The validity flag associated with the parameter is updated accordingly. If valid, the specified number of bits are fetched from the 32-bit ARINC-429 word and the scale factor is applied. The resultant value is then saved into the specified program variable. The linked list data structure is updated to store the time stamp associated with the input of the parameter. Note the time stamp is the value of the Condor ARINC-429 on-board clock saved by the on-board processor when the parameter was received.

The time-out value stored in the linked lists is used to invalidate parameters if they have not been received within the specified time-out period. The task *background* performs time-out operations. Each cycle of the task, it reads the Condor on-board clock and compares it to the time of last input for all parameters on all channels. This comparison determines the age of the data parameters. If the age of a data parameter exceeds the time-out limit stored in the linked list database, the parameter is invalidated.

Systran SCRAMNet+

The SCRAMNet VME card in the I/O Concentrator chassis includes the chassis into the aircraft's SCRAMNet ring. The SCRAMNet board is an intelligent I/O device with its own on-board processor. The applications software running on the MVME-1604 communicates with the SCRAMNet processor by addressing memory mapped control registers associated with the board. The board also provides 2 Megabytes of networked memory for shared data parameters within the VME address space. The control registers occupy E000-E03F of A16 space and the shared memory resides at 08400000-085FFFFF of A32 space. Detailed information about the SCRAMNet board may be obtained using the following reference.

Systran Corporation, support@systran.com

The SCRAMNet board is configured by the initialization task *run*. During initialization the SCRAMNet control registers are set up to insert the chassis into the ring in the *burst* mode. All of the shared memory is cleared as well as the Auxiliary Control Ram (ACR) bits associated with the memory addresses. Specific ACR locations are set to enable shared memory interrupts and triggers. Interrupting other nodes on the ring and generation of an external output trigger is enabled for writes to the "System Synchronization Pulse" (*system_sync* within the shared memory #3 structure). The interruption of other nodes is also enabled for the "datalink received" words (*SCRAM1intr/SCRAM2intr* within the shared memory #2 structure). The interruption of the PowerPC software is enabled for reception of the "downlink command" and the "ROTO block complete" words (*SCRAM3intr/SCRAM4intr* within the shared memory area #3). Pointer variables are initialized to the various addresses corresponding to the data structures shown in "Figure 6. SCRAMNet memory allocation". See also "Appendix B

LVLASO SCRAMNet Memory" for a description of the variables defined within LVLASO shared memory areas #1 through #3.

Offset[1]	Size[2]	Description
0000	176	LVLASO shared memory #1
00B0	1,892	LVLASO shared memory #2
0814	52	LVLASO shared memory #3
0848	4	25Hz system sync pulse
084C	4	DAS output heartbeat
0850	4	I/O concentrator heartbeat
0854	152	Record block #1 (smem #2)
08EC	56	Record block #2 (smem #3)
0924	100	Record block #3 (I/O VME)
0988	1.998Mb	Remaining memory

Total of 2,440 bytes used

Figure 6. SCRAMNet memory allocation.

The real-time operation of the SCRAMNet card is mostly transparent to the applications software of the MVME-1604. Variables, within the SCRAMNet shared address space, updated by the applications are automatically transmitted around the ring by the SCRAMNet board. Likewise variables updated by other nodes are automatically updated in the SCRAMNet on-board RAM. The variables that are computed by the I/O Concentrator processing software are placed on the ring by the task *t25hz*. The ARINC-429 inputs that only require decoding are placed directly onto the ring by the task *t50hz* (see "Condor ARINC-429"). The SCRAMNet activity not transparent to the applications software is the handling of externally generated SCRAMNet memory and error interrupts.

SCRAMNet generated interrupts cause activation of the interrupt handler *scramHandler*. This handler determines if the interrupt was caused by a network error or a memory update. When a network

[1] Offset value is the number of bytes (hexadecimal) past the beggining of the SCRAMNet+ shared memory buffer.

[2] The size value is the number of bytes (decimal) occupied by the block of data

error interrupt occurs the type of network error is ascertained from the on-board registers and network interrupts are disabled for one second. This disabling is done to eliminate the chance of a static network error condition monopolizing the PowerPC CPU. When an interrupt is generated by the update of a shared memory location, the interrupt handler determines the address of the updated memory location and passes it to the function *memAction*. This function uses the address of the memory interrupt to determine the appropriate response to the interrupt. Refer to "Section 3.0 Datalink Processing" and "Section 5.0 Data Recording" for descriptions of the actions performed when shared memory interrupts are received.

Datum Frequency and Time Processor

The Time and Frequency Processor (TFP) is used by the I/O Concentrator for synchronization to Universal Time Code (UTC). The TFP board receives an external IRIG-B time code signal as its reference. On-board oscillators are conditioned to the external time to provide a high resolution time reference. The TFP board is an intelligent I/O device with its own on-board processor. The applications software running on the MVME-1604 communicates with the TFP processor by addressing memory mapped control registers associated with the board. The control registers occupy E080-E0BF of A16 space. Commands to the on-board CPU are performed by writing a stream of command bytes to an on-board FIFO, then signaling that a command is ready via a control register. Detailed information about the TFP may be obtained using the following reference.

bc635VME/bc350VXI
Time and Frequency Processor
User's Guide

Datum Incorporated, http://www.datum.com

The TFP is initialized and setup by the task *run*, which resets the TFP board and commands the on-board CPU into its IRIG-B synchronization mode. Operationally, the TFP is only used by the I/O Concentrator task *t25hz* to sample the on-board clock every 40 milliseconds. The sampled time indicates the UTC marking the start of the associated iteration of the 25 Hz applications computations. The UTC is stored in SCRAMNet memory for use by other network nodes.

VMIVME Discretes

There are two discrete boards in the VME chassis, one for inputs and the other for outputs. The input discrete board is the VMIVME-1129 featuring 128 discrete inputs. The VMIVME-2128 board performs the output of 128 discretes. The applications software running on the MVME-1604 communicates with the discrete boards by addressing memory mapped control registers. The control registers occupy VME A16 space; E060-E07F for the VMIVME-1129, and E040 - E05F for the VMIVME-2128. The input discretes are connected to the Pilot Input Device (PID) switches and buttons. The output discretes control the two lights on the PID. Detailed information about the discrete boards may be obtained using the following references.

VMIVME-2128
128-bit High Voltage Digital Output
Board

VMIVME-1129
128-bit Digital Input Board
with Built-in-Test

VME Microsystems International Corp.; 1-800-322-3616
info@vmic.com

The discrete boards are initialized by the task *run*. To limit the number of VME accesses, the discrete boards are not accessed each time an individual discrete is referenced for either input or output. A

global data structure containing a bit for each of the I/O discretes is maintained within the MVME-1604 applications software. The input portion of the data structure is refreshed with the current state of the input discrete lines every 40 milliseconds by the task *t25hz*. Likewise, the bits in the data structure representing the state of the output discretes are output to the PID at the same rate by *t25hz*. When the applications software references or modifies a discrete via the *set_disc* and *test_disc* functions, the global data structure is what is actually accessed.

VMIVME Analog

Analog data is input to the I/O Concentrator via the VMIVME-3122 board. The applications software running on the MVME-1604 communicates with the analog board by addressing memory mapped control registers. The control registers occupy the addresses F000 - FFFF of VME A16 space. The board is capable of supporting 64 input analog channels, and is set up to perform "Auto-Scan" inputting of 32 channels. Each channel's analog input is a voltage in the range -10 to +10 volts. The digital conversion provides a 16 bit binary value in the range -32768 to 32767. Detailed information about the VMIVME-3122 may be obtained using the following reference.

VMIVME-3122
High Performance 16-Bit Analog to
Digital Converter Board

VME Microsystems International Corp.; 1-800-322-3616
info@vmic.com

The task *t50hz* inputs 6 analog channels every 20 milliseconds. The last three are only used for observation on the console monitor. The first three are connected to the body mounted accelerometers which are optionally used in GPS navigation calculations. The remaining channels are not used in the LVLASO configuration.

Section 3.0 Datalink Processing

The LVLASO datalinks are used for communications with airport ground stations. There are three datalinks used; two uplinks and one downlink. The downlink and one of the uplinks is used for two-way communications with Air Traffic Control (ATC). The other uplink is used to report airport status and traffic information to the NASA Boeing 757. The I/O Concentrator serves as a central point for datalink communications. It receives raw datalink transmissions and gathers message packets. The actual message fields are extracted from the packets and formatted as needed by the LVLASO nodes on the SCRAMNet ring. Downlinks are initiated external to the I/O Concentrator on the SCRAMNet ring. The downlink message bytes are packaged into the packet protocol and transmitted to the ground station by the I/O Concentrator.

All of the datalink I/O between the I/O Concentrator and the ground station is ARINC-429 block mode transmission at 100k bps. The datalink interface between the I/O Concentrator and the LVLASO workstations is the SCRAMNet ring (see "Systran SCRAMNet+"). The ARINC-429 datalink inputs to the VME chassis are managed by the Condor ARINC-429 board. The datalink words are input like the single ARINC-429 parameters obtained from the airplane data buses (see "Condor ARINC-429"), they simply fill the Condor on-board circular queue as they are received and are extracted every 20 milliseconds by *t50hz*. The downlinks are managed in a non-standard fashion on the Condor ARINC-429 board. Normal Condor ARINC-429 outputting is performed repetitively by the Condor on-board processor. A Message Data Block (MDB) is allocated for each output parameter which contains the parameter's initial 32-bit ARINC output word and an update rate. The Condor on-board processor is notified to include the MDB in its output list and the data value in the MDB will be output repeatedly, at the specified rate. These steps are typically done only once, at system boot up. The applications software is responsible for modifying the data value in the MDB as necessary during normal execution. This scheme does not work for ARINC-429 block transfers since each word in the block must be sent exactly one time, and in the order defined by the block. The version 1.15 firmware revision of the Condor ARINC-429 board provided the non-standard technique for sending block 429 transfers. A set of 200 MDBs are allocated at system boot up, however they are not included in the Condor on-board processor's output list initially. When a downlink block is ready for transmission, each word in the block is stored into a different reserved MDB and the MDB rate parameter is set to zero. The zero repeat rate indicates a one time send to the Condor on-board processor. Then the Condor processor is commanded to include each MDB individually, in the order required by the data block. A single output of each of the words in the block is achieved in this fashion. This technique is significantly less efficient than the standard mode since the Condor on-board processor must be commanded once for each word of every block transmitted, and these commands are generated during normal I/O Concentrator operations as opposed to system boot up.

ATC Uplinks

ATC uplinks are used to notify the flight crew of clearances from the airport tower. They arrive at the I/O Concentrator as ARINC-429 words and are passed onto the SCRAMNet as ASCII text strings. The ATC uplinks are received in channel #13 of the Condor ARINC-429 board. This channel is configured to 'listen' to ARINC label 104 (any SDI field) by the task *run*. Whenever a word is extracted from the queue associated with this channel, it is passed to the ATC uplink handler. This function enforces the rules of the ATC uplink protocol[3]. It gathers the words belonging to a single uplink block while verifying each word's role in the packet protocol. Message data bytes (DO-219 code) are extracted from the uplink data words as they are processed. Once all the message data bytes for an uplink are obtained, a pointer to the complete DO-219 message is placed into a VxWorks Message Queue to await handling by the task *atcRecv*. When an error in the uplink protocol is found, the started message is discarded and a NULL pointer is placed on the Message Queue. *atcRecv* normally is dormant, not using any PowerPC processor resources. The operating system awakens the task when a message is placed in its Message Queue. The pointer stored in the Message Queue is fetched by *atcRecv* and used to access the DO-219 byte stream stored in memory. If the passed pointer was NULL the parsing is skipped and SCRAMNet is updated immediately with the error

[3] Refer to the Controller-Pilot Data Link Communications Messages Document, Revision 7, dated May 9, 1997 for detailed information on ATC uplink protocol.

status. The DO-219 message protocol[4] governs the decoding of the data bytes. The byte stream is treated as a sequence of bit fields. The initial bit field controls which other bit fields must be defined in the message. Parsing of the bit fields continues until either an erroneous situation is detected or the complete message is ascertained. When the parsing is finished the resultant ASCII text string is placed on the SCRAMNet ring, then the "uplink received" SCRAMNet word is incremented to cause an interrupt of the LVLASO SGI nodes. The interrupt informs the nodes that a datalink was received. A SCRAMNet Boolean is used to signify error status of the received message.

There are a set of variables shown on the console monitor which are associated with the ATC uplinks. The following list describes the meaning of these variables.

atc_good This variable is the count of valid ATC uplinks passed to the LVLASO nodes since the boot up of the system.

atc_sync This variable counts the number of ARINC-429 words received out of synchronization. These words are discarded. A word out of synchronization is any word received while not currently building a packet (other than the word that denotes the beginning of a packet).

sequence_warn This variable counts the number of packets received which did not have a different sequence number than the previous packet.

atc_err This variable is the count of ARINC-429 words discarded because of uplink protocol violation. When a word is received which does not match the protocol, all previously received words in the packet are also discarded and included in this count.

atcDataErr This variable is the count of ATC uplink messages rejected by the task *atcRecv* because of errors detected during DO-219 parsing.

uplerr This is an error code which can be set by the task *atcRecv* while parsing the DO-219 bytes. The error code is set for each message received (good equals zero), therefore the code will be lost when a new message is received. The following is a list of the codes and their meaning.

1	Compound message received, not allowed for LVLASO implementation.
2	Exhausted data bits while accessing *reference number* available flag.
3	Exhausted data bits while fetching *message ID* number.
4	Exhausted data bits while fetching *message reference number.*
6	Exhausted data bits while fetching *message element ID* or invalid ID.
10	Exhausted data bits while accessing *altimeter format* code or code invalid.
11	Exhausted data bits while fetching *altimeter* value or value invalid.
34	Exhausted data bits while fetching *cross position* flag or *expedite cross* flag.
70	Exhausted data bits while fetching frequency type or type not VHF.
71	Exhausted data bits while fetching frequency value or value out of range.
80	Exhausted data bits while fetching *facility ID* flag.
81	Exhausted data bits while fetching *facility ID* type.
82	Exhausted data bits while fetching character count in *facility name.*
83	*Facility name* conversion error.
84	*Facility ID* conversion error.
85	Exhausted data bits while fetching *facility function.*
110	Exhausted data bits while fetching *position to hold* type or type not valid.
150	Exhausted data bits while fetching *ramp* number or number out of range.
151	Exhausted data bits while fetching *ramp direction.*
190	Exhausted data bits while fetching *runway direction* or direction invalid.
191	Exhausted data bits while fetching *runway configuration.*
200	Exhausted data bits while fetching *taxi route* type.
210	Exhausted data bits while fetching *canned taxi route.*
211	Exhausted data bits while fetching *canned taxi route* number.
220	Exhausted data bits while fetching *taxiway ID* or ID invalid.

[4] Refer to the Controller-Pilot Data Link Communications Interface Control Document, Revision 1.0, dated March 19, 1997 for detailed information on DO-219 message protocol.

230 Exhausted data bits while fetching *segment count* for taxiway list.
231 Exhausted data bits while fetching *taxiway segment ID.*
232 Exhausted data bits while fetching *segment ID flag* included discrete.
4096 Specified bits not available or character not alpha-numeric.
8192 Attempted to extract more data then sent with the message.

ATC Downlinks

ATC downlinks typically occur as an acknowledgment to received ATC uplinks, however they can also be initiated by on-board activities. In all cases, they are initiated by the LVLASO nodes on the SCRAMNet ring. The job of the I/O Concentrator is to recognize which downlink is being attempted, generate the corresponding DO-219 message stream, and wrap the message in the ARINC-429 block downlink protocol[5].

The LVLASO nodes store a message type code and an acknowledgment identifier in SCRAMNet memory, then interrupt the MVME-1604 via a SCRAMNet memory interrupt. Depending on the type of message, an optional runway label may also be stored into the SCRAMNet memory. The code identifies which of the messages is to be downlinked. If the downlink is an acknowledgment to a previous uplink, the identifier will specify which one. The SCRAMNet memory interrupt handler running on the PowerPC (*memAction*) extracts the data stored by the LVLASO nodes and inserts it in a VxWorks Message Queue. The operating system detects the presence of data in the Message Queue and awakens the dormant task *atcXmit.* This task fetches the data out of the Message Queue and generates the appropriate DO-219 byte stream. It then inserts the byte stream into the ARINC-429 downlink protocol and places the entire message into the Condor ARINC-429 transmitter queue.

There are a set of variables shown on the console monitor which are associated with the ATC downlinks. The following list describes the meaning of these variables.

down_good This variable is a count of downlinks completed without error.
down_err This variable is a count of the downlinks rejected because of an error in the data stored by the LVLASO nodes on the SCRAMNet.
dwlerr This variable is a code that describes the reason for the rejection of downlink data stored into SCRAMNet memory by the LVLASO nodes. The error code is set for each message sent (good equals zero), therefore the code will be overwritten by the next downlink. The following is a list of the bits within the code, and their meaning.

Bit #0 *Message reference number* out of range.
Bit #1 *Message reference number* for taxiway related downlinks not zero.
Bit #2 *Taxiway ID* not an uppercase letter.
Bit #3 Invalid *message element ID.*
Bit #4 *Taxiway segment ID* not a decimal number.

AMASS/Traffic Uplinks

AMASS uplinks are used to update the LVLASO map display with current aircraft conditions and traffic. They arrive at the I/O Concentrator as ARINC-429 words and are passed onto the SCRAMNet as blocks of structured data. The AMASS uplinks are received in channel #12 of the Condor ARINC-429 board. This channel is configured to 'listen' to ARINC labels 45 and 46 (SDI = 0) by the task *run.* The label 350 is also armed, but is not part of the AMASS packets. It is a word which relates to the strength of the radio signal and is only used for diagnostics.

Whenever a word is extracted from the queue associated with the AMASS uplink channel, it is passed to the AMASS uplink handler. This function enforces the rules of the AMASS uplink protocol[6]. It

[5] Refer to the Controller-Pilot Data Link Communications Messages Document, Revision 7, dated May 9, 1997 for detailed information on ATC downlink protocol.
[6] Refer to the GSI to VHF Radio Interface Document dated November 19, 1996 for detailed information on AMASS uplink protocol.

gathers the words belonging to a single uplink data packet while verifying each word's role in the packet protocol. It collects all the packets associated with a single radar scan of the airport, which is repeated every one second. There are two types of packets; airport status and target data. The fixed length airport status packet is always sent once at the beginning of a scan. Its reception by the I/O Concentrator marks the beginning of a radar scan. Following the airport status packet may be any number of target data packets. These packets will contain one or more data blocks containing position and identification information about a target aircraft at the airport. The AMASS uplink handler will continue to wait for target packets until the next airport status packet is received, indicating a new scan is starting. The data obtained from the previously received packets is formatted into structure blocks and the address of the completed set of data is placed in a VxWorks Message Queue. The operating system detects the presence of data in the Message Queue and awakens the dormant task *amassRecv*. This task fetches the buffer address from the Message Queue and uses it to parse the structured data fields. The airport information and traffic data blocks are stored in SCRAMNet memory and the related SCRAMNet interrupt word is incremented to inform the LVLASO nodes of the availability of a new airport scan.

During reception of an AMASS/Traffic uplink an error may be detected by the I/O Concentrator. In this situation the entire scan is discarded, however the SCRAMNet interrupt is still incremented after an AMASS error flag is set. This informs the LVLASO nodes that a new scan did appear, but had error status.

There are a set of variables shown on the console monitor which are associated with the AMASS uplinks. The following list describes the meaning of these variables.

amass_good This variable is a counter of all the valid AMASS/Traffic scans passed along to the LVLASO nodes on the SCRAMNet ring.

amass_sync This variable counts the number of ARINC-429 words received out of synchronization. These words are discarded. A word out of synchronization is any word received while not currently building a packet (other than the word that denotes the beginning of a packet).

amass_err This variable is the count of AMASS/Traffic scans discarded due to errors in the uplink packet protocol.

amassDataErr This variable is the count of AMASS/Traffic scans discarded due to errors in the embedded data. These errors are caused by unreasonable data parameters.

amasserr This variable is a code that describes the reason for the rejection of AMASS/Traffic uplinks due to unreasonable content. The error code is set for each message sent (good equals zero), therefore the code will be overwritten by the next uplink. The following is a list of the bits within the code, and their meaning.

Bit #0 *Runway wind speed* greater than 100 knots.
Bit #1 *Runway wind direction* greater than 359 degrees.
Bit #2 *Target aircraft latitude* position greater than 90 degrees.
Bit #3 *Target aircraft longitude* position greater than 180 degrees.

Section 4.0 GPS Processing

GPS position data from the Collins GPS receiver is input to the VME chassis at 1 Hz and passed through a 3rd order complementary filter to produce GPS derived position (in degrees of latitude and longitude, and feet of altitude) at 25 Hz. This filter is initialized to the Inertial Reference Unit (IRU) velocity and acceleration values at startup of the system. If GPS data is not available, the IRU position is used until GPS data becomes available. Subsequently, the filter is driven by accelerations, which are input at 50 Hz and passed through an alpha-beta filter having a 7 Hz cutoff frequency. These accelerations may be input from either the IRU or the experimental system body mounted accelerometers depending on operator selection. If the IRU accelerometers are in use, the acceleration vector is first corrected to compensate for the location of the IRU, which is well forward of the aircraft center of gravity (CG).

Once the filter is initialized, each input of GPS data is saved and propagated forward (at 25 Hz) using the velocity estimates. Each subsequent input is compared to the propagated value of the previous input, and rejected if it differs by more than a preset limit (100 meters). If the data is valid (as indicated by the validity bits) and passes the limit test, it is differenced with the saved value of the filter position output corresponding to the age of the GPS data measurement. The difference vector [EX] is input to the complementary filter to correct the position estimate. The age of the data is determined by measuring the time between arrival of the Digital Time Mark (ARINC word 124, output by the Collins GPS receiver) and arrival of the complete set of GPS position information.

Note that the location of the position estimate is the CG of the aircraft. This is to avoid filter perturbation due to aircraft rotation about its axes. The mechanization adds the antenna vector to the filter estimate before saving it in the history buffer. The value of [EX] is then the difference between the GPS measured position of the antenna and the filter estimate of the position of the antenna at the indicated point in time.

Besides GPS latitude, longitude, altitude, and measurement time, inputs include the GPS status word, the Horizontal and Vertical Dilution Of Precision words (HDOP and VDOP), and the validity bits for each of these words. All data must be valid and the GPS status word must indicate the receiver is functioning properly before any data is used. The requisite IRU data (Euler angles and accelerations) also must be valid before it can be used.

With GPS data valid, Differential GPS available (DGPS), and an acceptable HDOP and VDOP, the filter is checked for convergence. Once the average length of the difference vector [EX] has remained below 30 feet for 15 seconds, the flag *gpsVld* is set. This flag remains set as long as valid data meeting further accuracy requirements continues to be received. The flag is cleared if DGPS is lost for more than 30 seconds, if no valid GPS data is received for 4 seconds, or if the exceedance (difference) value of any input parameter is greater than 100 feet for 5 seconds. If *gpsVld* is not set, the accuracy of the GPS position data may not be sufficient for guidance on runways and taxiways. A 4-state status word is set on SCRAMNet at *smem1->dgps.GPSstatus* with the following significance:

0 – Best solution status with DGPS available.
1 – Good solution status, DGPS not available.
2 – DGPS just reacquired, solution may not be optimal.
3 – Bad solution status.

Section 5.0 Data Recording

The I/O Concentrator is responsible for managing data buffers for the Data Acquisition System (DAS). The DAS is the research system's real-time data recording facility which is used for post flight analysis of the events occurring during data runs. The data recording buffers managed by the I/O Concentrator reside in the SCRAMNet shared memory blocks. The DAS, a node on the ring, accesses the data directly from SCRAMNet memory.

There are four blocks of SCRAMNet shared memory updated by the I/O Concentrator software. The I/O Concentrator notifies the DAS every 40 milliseconds of the completed recording buffers via the SCRAMNet "system synchronization" interrupt word. This interrupt occurs half way through the 25 Hz real-time cycle of the I/O Concentrator. The following sections describe each of the four recording buffers.

Buffer #1 - Ownship State Parameters

This block contains the set of parameters describing the state of the Boeing 757. The data is obtained from direct inputting of aircraft data busses and internal I/O Concentrator calculations. It resides at SCRAMNet offset zero and is 176 bytes in length. This buffer is accessed by the DAS and the LVLASO nodes on the SCRAMNet ring. They can share this buffer since no special handling of the buffer is required by the DAS. The parameters of this buffer are updated at varying rates. Some values are updated as fast as every 20 milliseconds, while others are stored only once every two seconds. Some of the data in this buffer is 64 bit IEEE floating point format data. Since the SCRAMNet is a 32 bit ring, care must be taken by nodes on the ring to not use mismatched pairs of data for the 64 bit parameters. This is achieved using the "system synchronization" interrupt, which occurs every 40 milliseconds. Both the DAS and the LVLASO nodes should only access the data within the 20 milliseconds following the interrupt.

Buffer #2 - Uplink Messages

This buffer contains copies of both the AMASS/Traffic and ATC uplink messages. It is a separate buffer from that used directly by the LVLASO nodes since special processing is required to format this buffer. The 152 byte buffer starts at SCRAMNet offset 0854 (hex) and the formatting of the buffer is performed by the task *t25hz*. Because of the length of the messages, most of this data is multiplexed into a small set of recording words to efficiently make use of DAS resources. The uplinks arrive slowly, so the recording of their contents can be performed over several "system synchronization" frames after receiving the data. When either type of uplink has been processed without error, the uplink tasks *amassRecv* and *atcRecv* set flags to notify the *t25hz* task that new uplink data is ready.

The AMASS/Traffic data, excluding all of the target aircraft data, is immediately copied into corresponding locations within the buffer when a new uplink arrives. The target aircraft data blocks are multiplexed into a set of three target aircraft structures. Each recording cycle after receiving a AMASS/Target uplink, three new targets will be written into the structures. This will repeat until the entire set of received targets has been recorded. Afterward, these target structures will be zeroed until a new set of targets has been received. A maximum of 75 targets can be recorded with this technique since new target data is received every second, the recording update occurs at 25Hz, and there are three target structures. The LVLASO imposed limit on uplink targets received in one airport scan is 54.

The ATC uplink data is handled similar to the AMASS/Traffic uplink. The ASCII string associated with the ATC uplink is multiplexed into two bytes. The flag set by *atcRecv* causes the recording software to begin recording the start of the received message. Every recording cycle two more characters are stored. This continues until the entire message has been recorded. After this the two bytes are zeroed until the reception of another ATC uplink. There is no fixed time duration between message sent by the Air Traffic Control, however the time is typically large since these are initiated by human action. The longest ATC uplinks have approximately 36 characters, so this recording technique requires that 0.72 seconds of time must elapse between ATC uplinks.

Buffer #3 - ROTO Parameters and Downlink Messages

This buffer contains copies of information stored into SCRAMNet memory by the LVLASO nodes for use by the I/O Concentrator. It starts at SCRAMNet offset 08EC (hex) and is 56 bytes in length. This buffer exists separate from the original data because of synchronization. The LVLASO nodes operate asynchronous to the I/O Concentrator, therefore they may store data into SCRAMNet memory at any time within the I/O Concentrator's 40 millisecond cycle. If the LVLASO nodes updated this data near the "system synchronization" interrupt, a mismatched set of data could be recorded by DAS.

When the LVLASO nodes update either ROTO parameters or downlink data a SCRAMNet interrupt is sent to the I/O Concentrator. The SCRAMNet memory interrupt handler *memAction* determines which type of update was performed (ROTO or downlink) and the associated data is copied to local memory. A flag is also set to notify the recording software that new data was fetched. When the recording software executes, early in the 40 millisecond frame, it copies the saved data to the DAS buffer as a complete set eliminating the synchronization problem. This data recording technique of asynchronous data from external SCRAMNet nodes will fail to record all information if the LVLASO nodes update their data faster than the recording cycle of the I/O Concentrator. Neither ROTO or downlink data is updated rapidly.

Buffer #4 - I/O Concentrator Parameters

A DAS buffer is set aside for the recording of I/O Concentrator internal variables. These variables reside in local RAM on the MVME-1604 PowerPC processor. The recording software copies them from local memory to the SCRAMNet buffer every 40 milliseconds. The following is a list of the variables recorded in this 25 (32-bit) word buffer, which starts at SCRAMNet offset 0924 (hex).

gpsStat	GPS status word from Collins receiver
vdop	GPS vertical dilution of precision from Collins receiver
hdop	GPS horizontal dilution of precision from Collins receiver
navcom.gpectr	Navigation counter of exceedance frames
navcom.gptctr	Navigation time-out counter
navcom.gprun	GPS validation sequence counter
navcom.ex	Navigation filter exceedance vector (X, Y, Z)
navcom.biasba	Navigation filter acceleration bias vector (X, Y, Z)
navcom.velhgp	Navigation filter velocity estimate vector (X, Y, Z)
sigStrength	The signal strength received from AMASS/Traffic uplink
net_nodes	Number of nodes inserted on the SCRAMNet ring
decode_err	Number of rejected input ARINC-429 words
not_flushed	Number of frames where ARINC-429 input queue was not emptied
dwlerr	Downlink error code (see "ATC Downlinks")
amasserr	AMASS/Traffic uplink error code (see "AMASS/Traffic Uplinks")
uplerr	ATC uplink error code (see "ATC Uplinks")
rebootCode	Reason for automatic reboot of I/O Concentrator

Figure 7. I/O Concentrator recorded variables.

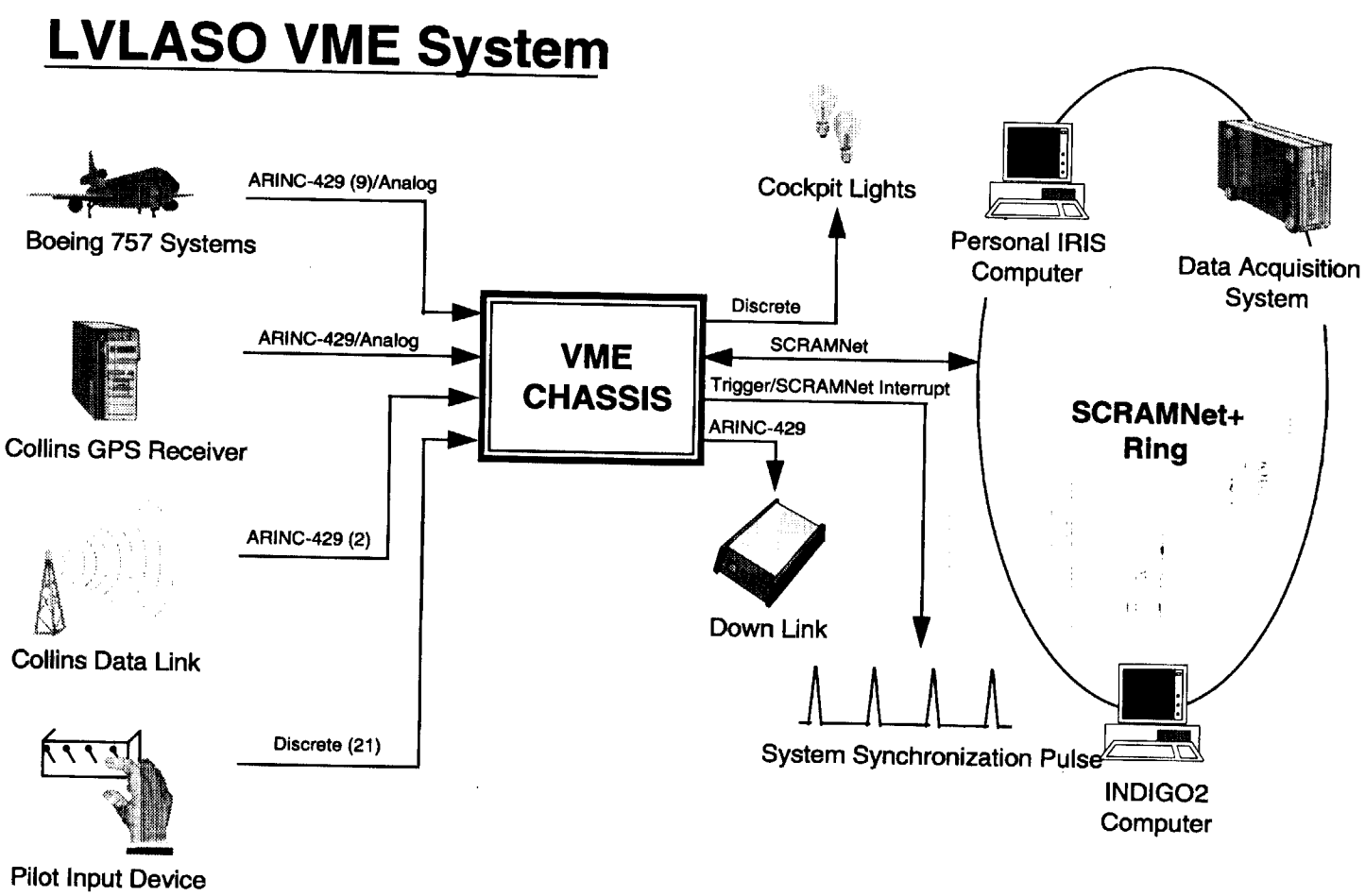


Figure 8. System connectivity.

Appendix A

ARINC-429 Receiver Channels

Channel	Source	Speed	ID
0	Air Data Computer bus #1	Low	6
1	<not used>		
2	Inertial Reference Unit bus #1	High	4
3	Flight Management Computer bus #4	High	2
4	EICAS bus #1	High	29
5	Global Positioning Satellite	High	NA
6	RADAR Altimeter bus #1	Low	7
7	Thrust Management Computer bus #4	Low	2A
8	Instrument Landing System bus #1	Low	10
9	Flight Controls Computer bus #1	Low	1
10	Flight Management Computer bus #2	Low	2
11	<not used>		
12	AMASS datalink	High	NA
13	ATC datalink	High	NA
14	<not used>		
15	<not used>		

Figure 9. ARINC-429 receiver channels.

Description	Label	SDI	Bits	Resolution	Rate (SPS)	Channel
Calibrated Airspeed	206	Any	15	.0625 Knots	8	0
True Airspeed	210	Any	16	.0625 Knots	8	0
Total Air Temperature	211	Any	12	.25 Deg C	2	0
Barometric Altitude	204	Any	18	1.0 Foot	16	0
Wind Speed	315	0	9	1.0 Knots	10	2
Heading (true)	314	0	16	.005493164 Deg	20	2
Pitch	324	0	15	.010986328 Deg	50	2
Roll	325	0	15	.010986328 Deg	50	2
Track Acceleration	362	0	13	.0009765625 G's	50	2
Cross Trk Acceleration	363	0	13	.0009765625 G's	50	2
Vertical Acceleration	364	0	13	.0009765625 G's	50	2
Track Angle (true)	313	0	16	.005493164 Deg	20	2
Wind Direction (true)	316	0	16	.005493164 Deg	10	2
Pitch Rate	326	0	14	.015625 Deg/Sec	50	2
Roll Rate	327	0	14	.015625 Deg/Sec	50	2
Yaw Rate	330	0	14	.015625 Deg/Sec	50	2
Longitudinal Acc.	331	0	13	.0009765625 G's	50	2
Lateral Acceleration	332	0	13	.0009765625 G's	50	2
Normal Acceleration	333	0	13	.0009765625 G's	50	2
Flight Path Acc.	323	0	13	.0009765625 G's	50	2
Vertical Speed	365	0	16	1.0 FPM	50	2

Figure 10. ARINC-429 inputs.

Description	Label	SDI	Bits	Resolution	Rate (SPS)	Channel
North Velocity	366	0	16	.125 Knots	10	2
East Velocity	367	0	16	.125 Knots	10	2
Flight Path Angle	322	0	16	.005493164 Deg	20	2
Ground Speed	312	0	16	.125 Knots	20	3
Latitude	310	N/A	21	1.716614e-4 Deg	5	3
Longitude	311	N/A	21	1.716614e-4 Deg	5	3
Rudder Position	033	0	12	.087890625 Deg	10	4
Elevator Position	017	0	12	.087890625 Deg	10	4
Air/Ground	015	0	1	Bit 15	0.5	4
Nose Wheel	012	0	1	Bit 20	0.5	4
Reverse Thrust	011	0	2	Bit 14,28	0.5	4
GPS Status	273	All	19	Bit String	1	5
GPS Altitude	076	N/A	21	.125 Feet	1	5
GPS HDOP	101	0	16	.03125 Units	1	5
GPS VDOP	102	0	16	.03125 Units	1	5
GPS Latitude Coarse	110	N/A	21	(see next)	1	5
GPS Latitude Fine	120	0	11	8.381903e-8 Deg	1	5
GPS Longitude Coarse	111	N/A	21	(see next)	1	5
GPS Longitude Fine	121	0	11	8.381903e-8 Deg	1	5
GPS Time Sync	124	All	N/A	N/A	1	5
GPS Final Word	141	All	N/A	N/A	1	5
Radar Altitude	164	All	17	.125 Feet	20	6
Throttle Position (left)	134	2	13	.04394531 Deg	5	7
Engine EPR (left)	340	2	13	.0009765625	5	7
Engine EPR (right)	340	1	13	.0009765625	5	7
Localizer Deviation	173	0	13	.976563e-4 DDM	16	8
Glideslope Deviation	174	0	13	.195313e-3 DDM	16	8
ILS Frequency	033	0	18	BCD[7]	5	8
Roll Command	140	All	13	.043945313 Deg	16	9
Pitch Command	141	All	13	.043945313 Deg	16	9
Go Around (pitch)	274	All	1	Bit 27	10	9
Go Around (roll)	275	All	1	Bit 20	10	9
Aircraft Weight	075	0	16	40 pounds	1	10

Figure 11. ARINC-429 inputs (continued).

[7] 4 bit hundredth, 4 bit tenth, 4 bit ones, 3 bit tens, 3 bit hundreds

Appendix B

LVLASO SCRAMNet Memory

The following is the structure definitions used in SCRAMNet memory to communicate between the I/O Concentrator and the LVLASO nodes on the SCRAMNet ring. These definitions are contained in the "header file" *scram.h*. It occupies SCRAMNet offsets 0000 - 084B (hex).

```
#define MAX_TARGETS 55

/*Shared Memory Area 1 - 168 bytes + 4 bytes for valids*/

struct FMS_Dat {

double      y_lat;          /*latitude; IRS or blended IRS/DGPS*/
double      x_long;         /*longitude; IRS or blended IRS/DGPS */
float       UTC_time;       /*time in millisecs past midnight GMT*/
short       radarAlt;       /*radar altitude in feet AGL*/
short       DGPSalt;        /*DGPS altitude in feet MSL*/
short       gSpeed;         /*ground speed in knots*/
short       vertSpeed;      /*vertical speed in knots*/
float       heading;        /*deg true north; CW from north is pos*/
float       yawRate;        /*deg/sec; nose right is pos*/
float       pitch;          /*deg; up is pos*/
float       pitchRate;      /*deg/sec; up is pos*/
float       roll;           /*deg; right wing down is pos*/
float       rollRate;       /*deg/sec; right wing down is pos*/
float       trackAng;       /*track angle; deg true; CW from north=pos*/
float       lonAccel;       /*in G's (32.2ft/sec**2); forward is pos*/
float       latAccel;       /*lateral acceleration in G's; right is pos*/
float       vertAccel;      /*in G's; up is pos*/
float       trackAccel;     /*in G's; forward is positive*/
float       crossTrkAccel;  /*in G's; right is pos*/
float       rightEngEPR;    /*right engine EPR; ratio always positive*/
float       leftEngEPR;     /*left engine EPR; ratio always positive*/
short       trueAirSpd;     /*true air speed in knots*/
short       calAirSpd;      /*calibrated air speed in knots*/
short       windSpd;        /*wind speed in knots*/
short       windDir;/*wind dir in deg true; CW from north=pos*/
short       totalAirTemp;   /*degrees C*/
short       throttlePos;    /*thrust degrees; forward thrust is pos*/
short       rudderPos;      /*degrees; right rudder is pos*/
short       elevatorPos;    /*degrees; surface up is positive*/
short       air_ground;     /*air ground system; discrete 0 or 1; main gear on ground = 1*/
short       noseWheelSquat; /*nose wheel squat system; discrete 0 or 1; nose wheel on ground = 1*/
float       acWeight;       /*weight of aircraft in lbs.*/
float       pitchFltCmd;    /*pitch flight director command*/
float       rollFltCmd;     /*roll axis (lateral) flight director cmd*/
float       fltPathAng;     /*flight path angle in degrees (+180 to-180)*/
float       fltPathAccel;   /*flight path acceleration in Gs (-4 to +4)*/
float       ILSfrequency;   /*ILS frequency*/
float       ILSlocDev;      /*ILS localizer deviation (+0.4 to -0.4 ); DDM*/
float       glideSlopeDev;  /*glide slope deviation (+0.8 to -0.8);DDM*/
short       ILScaptureFlag; /*ILS frequency captured; discrete*/
```

```
short           baroAlt;        /*corrected baro altitude in feet MSL*/
short           GoAround;       /*0 or 1; 1 = Go Around operative*/
short           FMSreserve1;
unsigned int    FMSvalids;      /* bit map of FMS valids*/
};

struct GPS_Dat {

float           lat;            /*GPS or DGPS latitude*/
float           lon;            /*GPS or DGPS longitude*/
short           GPSaltitude;    /*GPS altitude in feet*/
short           GPSreserve1;    /*can also get trk hdg,eastvel,northvel*/
unsigned int    GPSstatus;      /*GPS status word; 1=bad, 0=good*/
};

struct rtdata_aircraft {

unsigned int    inputFlags;     /*pilot input control switches*/
                                        /*used in simulation*/
short           opMode;
short           runNum;         /*used in simulation*/

struct FMS_Dat  ownship;        /*Flight Management System (FMS) data*/
struct GPS_Dat  dgps;           /*Global Positioning System (GPS) data*/
};


/*Shared Memory Area 2 - 1892 bytes*/

struct traffic_Dat {

int             id;             /*mode-S id or unique # for each target*/
char            fltNum[12];     /*flight number*/
float           y_lat;          /*y-coord or latitude*/
float           x_long;         /*x-coord or longitude*/
short           altAGL;         /*altitude AGL*/
short           heading;        /*course heading or track*/
short           gSpeed;         /*ground speed*/
short           reserve;        /*reserved*/
};

struct rtdata_ground {

    /*Traffic datalink - VHF */
    /*Traffic-target data*/
struct traffic_Dat target[MAX_TARGETS]; /*data for each target*/
int             numTargets;             /*# of targets each transmission*/
int             targetScanCnt;          /*a scan of traffic data is received*/

    /*AMASS data*/
unsigned int    AMASS_Hbars[4];         /*AMASS holdbars bit map*/
short           rwyWindSpd;             /*runway wind speed in knots*/
short           rwyWindDir;             /*runway wind direction in degrees*/
short           rwyFrictCoef;           /*not used - rwy friction coefficient*/
short           rwyRVR;                 /*runway visual range in feet*/
unsigned int    VHF1status;             /*VHF datalink 1 health;1=corrupted,0=good*/
```

```
unsigned int    SCRAM1intr;          /*SCRAMNET interrupt word -  VHF memory area */

    /*Controller datalink - Mode S */
short           msgNum;              /*msg number from controller message*/
short           msgType;             /*msg type from controller message*/
char            message[80];         /*controller uplinked message*/
unsigned int    modeSstat;           /*Mode S health; 1=corrupted,0=good*/
unsigned int    SCRAM2intr;          /*SCRAMNET interrupt word - Mode S memory area  */
};


/*Shared Memory Area 3 - 56 bytes including system_sync which formerly was the next
  4 bytes beyond the LVLASO specific SCRAMNET definition */

struct  lvlaso_outputs {

short           ATC_msgNum;   /*ATC msg being ACKed (0 if element id = 202 or 203*/
short           ATC_elementID; /*CPDLC downlink msg; 1=UNABLE,
                                 3=ROGER,202=TAXI DEVIATION,
                                 203=TURNED OFF ON TAXIWAY*/
char            taxiway[4];     /*Taxiway turned onto during ROTO (element id=203)*/
unsigned int    SCRAM3intr;     /*SCRAMNET interrupt word - Mode S downlink memory area */
int             PIDstates;      /*Pilot input device MSG and CLR states*/
float           velCmd;         /*ROTO velocity command*/
float           velDev;         /*ROTO velocity deviation*/
float           latPathDev;     /*ROTO lateral path deviation*/
short           rotoKnob;       /*ROTO  knob selection*/
short           rotoAlgRWY;     /*ROTO rwy selection */
float           rotoRWY_x;      /*ROTO rwy x-coordinate*/
float           rotoRWY_y;      /*ROTO rwy y-coordinate*/
int             rotoSpare1;
int             rotoSpare2;
unsigned int    SCRAM4intr;     /*SCRAMNET interrupt word - ROTO datablock memory area */
unsigned int    system_sync;    /*SCRAMNET synchronization interrupt word - lat/longs intact */
};
```

Appendix C

Using the *Monitor* Program

The task *monitor* is a utility program used to display the contents of I/O Concentrator global variables on the Console screen. This includes both variables in MVME-1604 PowerPC local RAM and shared SCRAMNet memory. There are 8 predefined pages of display variables containing the following information.

Page 1	LVLASO Shared memory #1 Ownship data
Page 2	Page #1 continued
Page 3	I/O Concentrator general diagnostics
Page 4	Datalink diagnostics
Page 5	GPS input parameters
Page 6	GPS processed parameters
Page 7	ATC decoded uplink message
Page 8	AMASS/Traffic decoded uplink message

Figure 12. Monitor display pages.

To use the program, the command "*monitor*" is entered at the shell prompt (either remote host or target shell). The program defaults to page #1 when started. To change to another page, enter the number of the desired page at the *monitor* prompt. The program is exited by entering "*e*" at the *monitor* prompt.

The variables shown on the various pages of the *monitor* are predefined in the file *dspinit.c*. To customize existing display pages or add/remove entire pages a new *dspinit.c* must be generated. This can be done by editing the file directly, however a utility program exists which makes the job easier. The utility *gen_dspinit* reads a page description file and writes a '.c' source file in the format of *dspinit.c*. The page description file has a series of "PAGE" commands which define, in order, the pages of the *monitor* program. The "PAGE" command also contains the page title which will be seen on the *monitor* display. Following the page command is entries for the variables, up to 20, which are to be shown on that page. There are four commands which are used to describe a variable intended for display. The commands are listed as follows.

NAME Specifies the variable's descriptive tag to be displayed on the monitor to the left of its value. This is a required entry.

TYPE Denotes the variable's type (float, signed int, etc.). This is also a required entry. The two types for ASCII text, *ASCII_TEXT* and *ASCII_LONG*, determine how text is displayed on the monitor. *ASCII_TEXT* causes text to be displayed in the monitor's standard double column framework. *ASCII_LONG* causes text to be displayed across the entire length of the monitor. When *ASCII_LONG* is used, all items on the same page are displayed in a single column. *ASCII_TEXT* and *ASCII_LONG* cannot be mixed on the same page.

LENGTH Indicates the length in bytes of the variable. The default is 4.

VARIABLE Identifies the actual software item from which the variable's displayed value is referenced. The reference point defaults to the value of NAME.

VALID This is an optional field. When present, it specifies a software flag to be checked to determine the validity of the variable's contents before displaying it. If the flag indicates an invalid condition, a bar is displayed on the monitor in place of the variable's value.

Observing the diagnostic variables during operations is a convenient way to oversee the health of the system. The diagnostic variables associated with the datalinks are describe in "Section 3.0 Datalink Processing". The following list explains the meaning of the remaining diagnostic variables.

verID This character string is the version identification of the I/O Concentrator software. For this release it will always be "**Error! Reference source not found.**".

good_429 This variable is the counter containing the number of ARINC-429 words received that were accepted as expected input and decoded without error.

decode_err This variable is the counter containing the number of ARINC-429 words received that were either not expected or caused a decode error.

not_flushed This variable is the counter which is incremented each time a Condor ARINC-429 receiver queue contains more than the maximum allowed (64) number of parameters which may be extracted in one 20 millisecond frame.

t50_over This variable is the count of the number of times the 50 Hz task exceeded its time frame of 20 milliseconds.

t25_over This variable is the count of the number of times the 25 Hz task exceeded its time frame of 40 milliseconds.

txTimeout This variable counts the number of times a SCRAMNet error interrupt was generated having the *transmitter time-out* bit set in the error register.

txRetry This variable counts the number of times a SCRAMNet error interrupt was generated having the *transmitter retry* bit set in the error register.

rcvOver This variable counts the number of times a SCRAMNet error interrupt was generated having the *receiver overflow* bit set in the error register.

bad_msg This variable counts the number of times a SCRAMNet error interrupt was generated having the *bad message* bit set in the error register.

no_carrier This variable contains the count of the number of times a SCRAMNet error interrupt was generated having the *no carrier bit* set in the error register.

protocol_err This variable counts the number of times a SCRAMNet error interrupt was generated having the *protocol error* bit set in the error register.

intFIFO_full This variable counts the number of times a SCRAMNet error interrupt was generated having the *interrupt FIFO full* out bit set in the error register.

txFIFO_full This variable counts the number of times a SCRAMNet error interrupt was generated having the *transmitter FIFO full* bit set in the error register.

net_nodes This variables contains the number of SCRAMNet nodes inserted onto the ring.

memIntRecv This variable contains the counts of the number of times the MVME-1604 PowerPC has been interrupted by external nodes on the SCRAMNet ring using memory interrupts.

false_intr This variable contains the count of the number of times the MVME-1604 PowerPC was interrupted by a SCRAMNet node using a memory address not armed for interrupting.

volts (0-5) These variables contain the input voltages for the first six of the analog input channels. The units are in volts.

discInRaw This variable contains the first 32 input discrete values.

discOutRaw This variable contains the first 32 output discrete values.

ROTOstatus This variable contains the status of the recording of the ROTO data block. The lower 16 bit field of the 32-bit value increments each time a ROTO block is fetched without failure. The upper 16 bit field of the 32-bit value is incremented each time a ROTO block can not be fetched because the previous block was not yet removed by data recording software.

Figure 13. *Monitor* diagnostics.

Appendix D

The SCRAMNet Test Pattern

During preflight testing the I/O Concentrator may be used to generate test data in the SCRAMNet buffers. This data is used by the Data Acquisition System (DAS) to verify the interface between the DAS and the I/O Concentrator. The "system test" task is separate from the standard I/O Concentrator software. If the standard applications tasks are running, they must be stopped before starting the test. Once the test software is running, the type of test may be selected using the shell prompt. The global variable *testNum* contains the identifier for which test pattern is used. The options are as follows.

testNum	test pattern
0	Incrementing pattern; each 32-bit word contains its offset value.
1	Shifting bit; each frame all words set to new shift pattern.
2	Static engineering values.

Figure 14. "systest" selections.

```
-> sp stop                    (if standard tasks active)
-> sp systest
-> testNum=#                  (defaults to 0)
```

Appendix E

Source Code File Organization

'C' Source Code Files

ACCFLT.C	Filtering of accelerations
ADC.C	VME Analog-to-digital conversion
AMASS.C	AMASS/Traffic uplink
ATC_DOWN.C	ATC downlink
ATC_UP.C	ATC uplink
BACK_MGR.C	Background task
CON429.C	VME ARINC-429 setup
DATAREC.C	Data recording
DEC429.C	ARINC-429 decoding
DISCRETE.C	VME discrete input/output
DSPINIT.C	*Monitor* display page setup
GETDATA.C	GPS input data snapshot
GLOBAL.C	Global variable definitions
GPSPRC.C	GPS navigation
LINK.C	ARINC-429 datalink packet protocol
MONITOR.C	Console memory monitor
RUN.C	I/O Concentrator initialization task
SCRAMNET.C	VME SCRAMNet+
SETRECV.C	ARINC-429 receiver control
SYSTEST.C	Preflight recording test
T25HZ.C	25Hz task
T50HZ.C	50Hz task
TFP.C	VME Time Frequency Processor
UTIL.C	Utility functions

Header Files

ADC.H	VME analog-to-digital conversion
ATC_link.h	Datalinks
CON429.H	VME ARINC-429
DEFINE.H	System constant definitions and user data types
DISCRETE.H	VME discrete input/output
DSPDEF.H	Console monitor display
EXTERN.H	Global variable 'extern' references
GLOBAL.H	Global function prototypes
IOCOM.H	GPS input variable structure template
IPDATA.H	CONDOR ARINC-429 definitions
NAVCOM.H	GPS navigation structure template
RECORD.H	I/O Concentrator recording table
SCRAM.H	VME SCRAMNet+ definitions
TFP.H	VME Time Frequency Processor definitions

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE
January 1998

3. REPORT TYPE AND DATES COVERED
Contractor Report

4. TITLE AND SUBTITLE
The LVLASO I/O Concentrator Software Description, Version 3.5

5. FUNDING NUMBERS
NAS1-20431
WU 522-14-31-02

6. AUTHOR(S)
Christopher J. Slominski, Valerie E. Plyler, and David A. Wolverton

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Computer Sciences Corporation
Systems Sciences Division
Hampton, VA 23666-1379

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
NASA Langley Research Center
Hampton, VA 23681-2199

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA/CR-1998-206907

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Lucille H. Crittenden

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 61 Distribution: Nonstandard
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)
This paper describes the software written for the I/O Concentrator Unit in support of the Low Visibility Landing and Surface Operations (LVLASO) experiment flown on-board NASA's Boeing 757 aircraft.

14. SUBJECT TERMS
Boeing 757, ARINC-429, Data Link, VxWorks, GPS

15. NUMBER OF PAGES
35

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	